

Constant-approximation algorithms for highly connected multi-dominating sets in unit disk graphs

Takuro Fukunaga

National Institute of Informatics, Japan

JST, ERATO, Kawarabayashi Large Graph Project, Japan

takuro@nii.ac.jp

Abstract

Given an undirected graph on a node set V and positive integers k and m , a k -connected m -dominating set $((k, m)$ -CDS) is defined as a subset S of V such that each node in $V \setminus S$ has at least m neighbors in S , and a k -connected subgraph is induced by S . The weighted (k, m) -CDS problem is to find a minimum weight (k, m) -CDS in a given node-weighted graph. The problem is called the unweighted (k, m) -CDS problem if the objective is to minimize the cardinality of a (k, m) -CDS. These problems have been actively studied for unit disk graphs, motivated by the application of constructing a virtual backbone in a wireless ad hoc network. However, constant-approximation algorithms are known only for $k \leq 3$ in the unweighted (k, m) -CDS problem, and for $(k, m) = (1, 1)$ in the weighted (k, m) -CDS problem. In this paper, we consider the case in which $m \geq k$, and we present a simple $O(5^k k!)$ -approximation algorithm for the unweighted (k, m) -CDS problem, and a primal-dual $O(k^2 \log k)$ -approximation algorithm for the weighted (k, m) -CDS problem. Both algorithms achieve constant approximation factors when k is a fixed constant.

1 Introduction

Compared with a traditional communication network, a wireless ad hoc network has the advantage of not requiring any infrastructure, such as base stations and WiFi routers; this is a great benefit when operating sensor networks, vehicle networks, or networks in disaster areas. However, for efficient operation of a wireless ad hoc network, we have to overcome many technical challenges. One of these is to reduce the redundant communication caused by flooding messages. Since it is difficult to maintain information over an entire wireless ad hoc network, when a message is sent from one node to another, it is common practice to flood the message to all nodes in the network. This uses more power than is necessary, and lowers the efficiency of the network.

A typical solution to the inefficiency due to flooding is to construct a *virtual backbone network*, as follows: we choose several backbone nodes from the network, and then we construct a subnetwork that comprises only these backbone nodes. When a message arrives, first, it is delivered to a backbone node, next, it is flooded to all the backbone nodes via the virtual backbone network, and finally, the destination node receives the message from an adjacent backbone node. This improves energy efficiency more as the virtual backbone network is smaller. However, it is also important that the virtual backbone network be fault tolerant.

Developing algorithms for constructing a virtual backbone network is an active area of research. A promising approach is to formulate a virtual backbone network as a *connected dominating set* (CDS), and to consider an algorithm for finding a minimum cardinality or a minimum weight CDS. For an undirected graph with a node set V , a CDS is defined as a subset S of V such that each node in $V \setminus S$ is adjacent to at least one node in S , and S induces a connected subgraph. This approach has gained in popularity, and many proposals have been published. It is typically assumed that the input graph is a unit disk graph, which is a

natural choice for modeling a wireless network. Since the problem of finding the minimum cardinality CDS is NP-hard even for unit disk graphs [8], some studies have considered approximation algorithms.

A CDS does not give a fault-tolerant virtual backbone network. This is because a CDS is only required to be connected, and each node outside a CDS is required to have only one neighbor in the CDS. Hence, if a backbone node fails, the virtual backbone network may be disconnected, or a non-backbone node may lose access to the virtual backbone network. To overcome this disadvantage, Dai and Wu [9] proposed replacing a CDS by a *k-connected k-dominating set*, and they addressed the problem of finding a minimum *k-connected k-dominating set* in a unit disk graph. A subset S of the node set V is called *k-connected* if the subgraph induced by S is *k-connected* (i.e., it is connected even if any $k - 1$ nodes are removed), and is called *k-dominating* if each node $v \in V \setminus S$ has k neighbors in S . Triggered by their study, much attention has been paid to this problem, extending the notion of a *k-connected k-dominating set* to a more-general *k-connected m-dominating set* ((k, m) -CDS).

The problem of finding a minimum cardinality (k, m) -CDS in a unit disk graph is called the *unweighted (k, m) -CDS problem*. If each node is given a nonnegative weight, and the objective is to minimize the weight of a (k, m) -CDS, then this is called the *weighted (k, m) -CDS problem*. As for the unweighted (k, m) -CDS problem, a constant-approximation algorithm was given by Shang et al. [17] for $k \leq 2$, and by Wang et al. [20] for $k = 3$. The latter algorithm was improved by Wang et al. [21]. We note that the analysis of the algorithm for $k = 2$ in [17] contains an error, and thus the approximation factor given therein is not correct; although this is not our main focus, it will be explained in Appendix A. As for the weighted (k, m) -CDS problem, there are several constant-approximation algorithms for $k = m = 1$, but we are aware of no previous studies that investigate the case of $(k, m) \neq (1, 1)$.

After these previous studies, a natural question arises as to whether there is a constant-approximation algorithm for the unweighted (k, m) -CDS problem with $k \geq 4$, and for the weighted problem with $(k, m) \neq (1, 1)$. For the unweighted problem, this question has been already addressed in both [20] and [21]. We answer this question affirmatively. The main contribution of this paper is to present constant-approximation algorithms for both the unweighted and the weighted (k, m) -CDS problems when k is a constant and $k \leq m$.

Specifically, we present two algorithms; one is an $O(5^k k!)$ -approximation algorithm for the unweighted (k, m) -CDS problem, and the other is an $O(k^2 \log k)$ -approximation algorithm for the weighted (k, m) -CDS problem. Notice that both algorithms achieve a constant factor if k is a constant. The approximation factor of the latter algorithm is better than that of the former, and it can be applied to the weighted problem, while the former algorithm is restricted to the unweighted problem. However, the former algorithm is simple, easy to analyze, and can also be applied to other graph classes. In fact, for $k \in \{2, 3\}$, the former algorithm is obtained by introducing more specification into the algorithms given in [17, 21]. Hence our analysis on the former algorithm gives a simple proof for the fact that the algorithms in [17, 21] achieve Constant-approximation for $k \in \{2, 3\}$.

Figure 1 shows a $(4, 4)$ -CDS computed by our $O(k^2 \log k)$ -approximation algorithm in a unit disk graph on 400 nodes. Square nodes represent the 161 nodes chosen to be part of the $(4, 4)$ -CDS.

1.1 Organization

The remainder of this paper is organized as follows. Section 2 surveys related works. Section 3 introduces preliminaries facts used in this paper. Section 4 presents our $O(5^k k!)$ -approximation algorithm for the unweighted (k, m) -CDS problem. Section 5 provides our $O(k^2 \log k)$ -approximation algorithm for the weighted (k, m) -CDS problem. Section 7 concludes the paper. Appendix A rectifies the analysis on the algorithm of Shang et al. [17] for the unweighted $(2, m)$ -CDS problem.

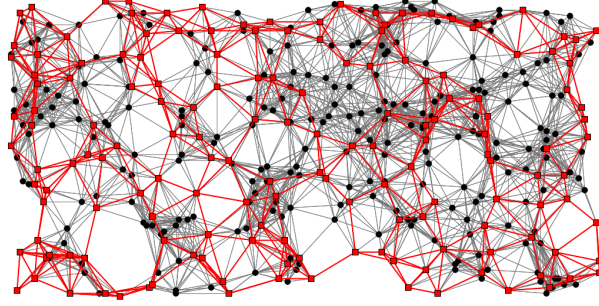


Figure 1: A $(4,4)$ -CDS computed by the $O(k^2 \log k)$ -approximation algorithm in a unit disk graph on 400 nodes

2 Related works

The study of the $(1,1)$ -CDS problem for general graphs was initiated by Guha and Khuller [13]. They presented an $O(H(\Delta))$ -approximation algorithm for the unweighted $(1,1)$ -CDS problem in graphs with maximum degree Δ , where $H(\Delta)$ denotes the Δ -th harmonic number. They also gave a reduction from the set cover to the unweighted $(1,1)$ -CDS problem, which shows that no polynomial-time algorithm achieves an approximation factor $(1-\epsilon)H(\Delta)$ for any fixed $\epsilon \in (0,1)$ unless $\text{NP} \subseteq \text{DTIME}[n^{O(\log \log n)}]$. For the weighted $(1,1)$ -CDS problem, they gave an $O(\log n)$ -approximation algorithm, where n is the number of nodes in the given graph.

The unweighted $(1,1)$ -CDS problem is NP-hard, even in unit disk graphs [8]. Marathe et al. [15] showed that the unweighted $(1,1)$ -CDS problem in unit disk graphs admits a 10-approximation algorithm. This has been improved by subsequent studies, and the current best result is due to Cheng et al. [5], who gave a polynomial-time approximation scheme (PTAS) for the unweighted $(1,1)$ -CDS problem in unit disk graphs; note that the existence of a PTAS means that for any fixed constant $\epsilon > 0$, there exists a $(1+\epsilon)$ -approximation algorithm that runs in polynomial time.

The first constant-approximation algorithm for the weighted $(1,1)$ -CDS problem in unit disk graphs was due to Ambühl et al. [2]. The current best approximation factor for the same problem is $(4+\epsilon)$. This is achieved by combining the $(3+\epsilon)$ -approximation algorithm due to Willson et al. [22] for the minimum weight 1-cover problem, and the $(1+\epsilon)$ -approximation algorithm due to Zou et al. [25]. for the node-weighted Steiner tree problem.

The first study of the (k,m) -CDS problem was conducted by Dai and Wu [9]. They found several heuristic algorithms for the unweighted problem with $k = m$ in unit disk graphs. Thus far, several constant-approximation algorithms have been given for the unweighted (k,m) -CDS problem in unit disk graphs. For example, the algorithm of Thai et al. [18] is for $k = 1$, the algorithm of Wang, Thai, and Du [19] is for $(k,m) = (2,1)$, and the algorithm of Wang et al. [20] is for $k = 3$. The current best result is due to Shang et al. [17] for $k \leq 2$, and due to Wang et al. [21] for $k = 3$. Several previous papers claimed constant-approximation algorithms for $k \geq 4$, but Kim et al. [14] showed that all of them had technical errors. As far as we know, there is no known constant-approximation algorithm for the unweighted (k,m) -CDS problem with $k \geq 4$ or for the weighted (k,m) -CDS problem with $(k,m) \neq (1,1)$. For the weighted problem, it is not even known whether a constant-approximation algorithm exists for the problem of finding a minimum weight m -dominating set in a unit disk graph when $m > 1$.

We note that distributed algorithms for the unweighted $(1,1)$ -CDS problem are also actively being studied. Since this paper will focus on centralized algorithms, we will only refer to a few of these studies [1, 11, 23].

3 Preliminaries

Let $\alpha \geq 1$. An algorithm for a minimization problem is called an α -approximation if it always outputs a feasible solution whose objective value is at most α times the optimal value for any feasible instance. If an algorithm is an α -approximation, α is called the *approximation factor* of this algorithm. When the approximation factor is a constant, the algorithm is called a *constant-approximation algorithm*.

Let $G = (V, E)$ be an undirected graph with a node set V and an edge set E . For $X \subseteq V$, let $G[X]$ denote the subgraph of G induced by X ; i.e., its node set is X and the edge set consists of the edges that join nodes in X . Throughout the paper, on the power set of V , we define maximality and minimality with respect to inclusion. In other words, X is minimal in a family $\mathcal{V} \subseteq 2^V$ if there is no $Y \in \mathcal{V}$ with $Y \subset X$, and X is maximal in \mathcal{V} if there is no $Y \in \mathcal{V}$ with $X \subset Y$.

In a *unit disk graph*, each node is placed on the two-dimensional Euclidean space, and two nodes are joined by an edge if and only if the distance between them is not larger than a unit length. The following property of unit disk graphs is well known, and it is used in [17].

Lemma 1. *Let $G = (V, E)$ be a unit disk graph. Let $v \in V$, and let u_1, \dots, u_6 be distinct neighbors of v . Then E includes an edge that joins two nodes in $\{u_1, \dots, u_6\}$.*

For the most part, our algorithms will require only the property stated in Lemma 1; the exception is the algorithm for computing a minimum weight m -dominating set, presented in Section 5.1.

For $X \subseteq V$, we denote the set of neighbors of X in G by $\Gamma(X)$. In other words, $\Gamma(X) = \{v \in V \setminus X : uv \in E \text{ for some } u \in X\}$. We also let X^+ denote $X \cup \Gamma(X)$ for any $X \subseteq V$. For $X, T \subseteq V$, we simply denote $\Gamma(X) \cap T$ by $\Gamma_T(X)$. The following property of the function Γ has been frequently used in previous works on the survivable networks design (see e.g., [16]). Here, we provide a proof for completeness.

Lemma 2. *For any $X, Y, T \subseteq V$, the following hold:*

$$|\Gamma_T(X)| + |\Gamma_T(Y)| \geq |\Gamma_T(X \cap Y)| + |\Gamma_T(X \cup Y)|, \quad (1)$$

$$|\Gamma_T(X)| + |\Gamma_T(Y)| \geq |\Gamma_T(X \setminus Y^+)| + |\Gamma_T(Y \setminus X^+)|. \quad (2)$$

Proof. We prove (1) by counting the contribution of each node $v \in T$ to each side. Let $v \in \Gamma_T(X \cap Y)$. Since v has a neighbor in each of X and Y , v belongs to either X or $\Gamma_T(X)$, and to either Y or $\Gamma_T(Y)$. If v is contained in both $\Gamma_T(X)$ and $\Gamma_T(Y)$, then $v \notin X \cup Y$ holds, and hence $v \in \Gamma_T(X \cup Y)$. Next, let $v \in \Gamma_T(X \cup Y)$. Then $v \notin X \cup Y$, and v has a neighbor in $X \cup Y$. If v has a neighbor in $X \cap Y$, then v is contained in both $\Gamma_T(X)$ and $\Gamma_T(Y)$, and $v \in \Gamma_T(X \cap Y)$ holds. Otherwise, v is contained in either $\Gamma_T(X)$ or $\Gamma_T(Y)$. In sum, if a node is counted at least once on the right-hand side of (1), it is also on the left-hand side. If the right-hand side counts a node twice, then the left-hand side does, too. Thus, (1) holds.

To prove (2), let Y' denote $V \setminus Y^+$. Then $\Gamma_T(Y) = \Gamma_T(Y')$, and $X \setminus Y^+ = X \cap Y'$ holds. In what follows, we prove that $Y \setminus X^+ = V \setminus (X \cup Y')^+$ holds. Since $\Gamma_T(V \setminus (X \cup Y')^+) = \Gamma_T(X \cup Y')$, this means that (2) can be proven by applying (1) to X and Y' .

$Y \setminus X^+ \subseteq V \setminus (X \cup Y')^+$ holds because the nodes in $Y \setminus X^+$ are not contained in $X \cup Y'$, and none has a neighbor in X (otherwise $v \in X^+$) or in Y' (otherwise Y' has a node in Y^+). Let $v \in V \setminus (X \cup Y')^+$. Then v is in neither X^+ nor $(Y')^+$, and $v \notin (Y')^+$ implies $v \in Y$. Hence, $V \setminus (X \cup Y')^+ \subseteq Y \setminus X^+$ holds. \square

Let $X \subseteq V$. For $T \subseteq V$, X is called a T -cut if $X \subseteq T$ and $X \neq \emptyset \neq T \setminus X^+$, and it is called a *Steiner T -cut* if $X \cap T \neq \emptyset \neq T \setminus X^+$. For $T \subseteq V$ and $r \in T$, a T -cut X (resp., a Steiner T -cut X) is called a (T, r) -cut (resp., Steiner (T, r) -cut) if $r \notin X^+$.

A graph G is k -connected when it is connected even when any $k - 1$ or fewer nodes are removed from the graph. We note that a graph on at most k nodes is k -connected by definition if it is a complete graph. By Menger's theorem, a graph G is k -connected if and only if $|\Gamma(X)| \geq k$ for any nonempty $X \subseteq V$ with $X^+ \neq V$. A subset T of V is k -connected if and only if $|\Gamma_T(X)| \geq k$ for any T -cut X (recall that the k -connectivity of T is defined by the k -connectivity of $G[T]$).

Our algorithms take the same approach as the algorithms proposed in previous studies; they compute an m -dominating set in the first step, and increase its connectivity by one in each of the subsequent steps. Hence, after computing an m -dominating set, our algorithms repeat the process to solve the following problem.

Definition 1 (Augmentation problem). *Given an undirected graph $G = (V, E)$, a nonnegative weight $w(v)$ for each node $v \in V$, and a $(k-1, m)$ -CDS $T \subseteq V$, find $S \subseteq V \setminus T$ that minimizes $\sum_{v \in S} w(v)$, subject to the condition that $T \cup S$ is k -connected.*

For $T \subseteq V$, a path P is called a T -path if both end nodes of P are included in T , and no inner nodes of P are included in T .

A family \mathcal{L} of subsets of V is said to be *laminar* if any $X, Y \in \mathcal{L}$ satisfies $X \subseteq Y$, $Y \subseteq X$, or $X \cap Y = \emptyset$. \mathcal{L} is said to be *strongly laminar* if any $X, Y \in \mathcal{L}$ satisfies $X \subseteq Y$, $Y \subseteq X$, or $X \cap Y^+ = \emptyset = X^+ \cap Y$. Let \mathcal{L} be a laminar family. If X is minimal in \mathcal{L} , we call X a *leaf* of \mathcal{L} . For some $Y \in \mathcal{L}$, if X is a maximal member of \mathcal{L} subject to $X \subset Y$, then we say that X is a *child* of Y .

4 Simple algorithm for the unweighted problem

In this section, we present an $O(5^k k!)$ -approximation algorithm for the unweighted (k, m) -CDS problem with $m \geq k$. We may assume G is k -connected, since otherwise G has no (k, m) -CDS. As noted above, our algorithm computes an m -dominating set by applying the constant-approximation algorithm given in [17], and it then increases the connectivity by repeatedly solving the augmentation problem.

Now let us assume that there is a $(k-1, m)$ -CDS T , and let us consider increasing its connectivity to k . Since T is $(k-1)$ -connected, all T -cuts X satisfy $|\Gamma_T(X)| \geq k-1$. We say a T -cut X is a *demand cut* if $|\Gamma_T(X)| = k-1$. We say that a T -path P *covers* a demand cut X if one end node of P is in X , and the other end node is in $T \setminus X^+$.

The following lemma was used in previous studies [17, 20, 21].

Lemma 3. *Let T be a $(k-1, m)$ -CDS, and let $m \geq k$. For every demand cut X , there is a T -path that covers X and contains at most two inner nodes.*

We also need the following fundamental lemma.

Lemma 4. *Let T be a $(k-1, m)$ -CDS, and let $m \geq k$. Let $S \subseteq V \setminus T$. If a $(T \cup S)$ -cut $X \subseteq T \cup S$ satisfies $|\Gamma_{T \cup S}(X)| \leq k-1$, then $X \cap T$ is a demand cut.*

Proof. Since T is an m -dominating set, if $X \subseteq S$, then $|\Gamma_{T \cup S}(X)| \geq |\Gamma_T(X)| \geq m \geq k$ holds, which contradicts the assumption. If $T \subseteq X$, then $X' := (T \cup S) \setminus X^+$ satisfies $k \leq |\Gamma_{T \cup S}(X')|$ by the same reason. Since $\Gamma_{T \cup S}(X') = \Gamma_{T \cup S}(X)$, this does not happen. Hence $X \cap T$ is a T -cut. The lemma is immediate from $\Gamma_T(X \cap T) \subseteq \Gamma_T(X) \subseteq \Gamma_{T \cup S}(X)$. \square

A node set $S \subseteq V \setminus T$ is feasible for the augmentation problem if every demand cut X is covered by a T -path in $G[S \cup T]$. To see this, suppose that $G[T \cup S]$ is not k -connected. Then there exists a $(T \cup S)$ -cut X such that $|\Gamma_{T \cup S}(X)| \leq k-1$. By Lemma 4, $X \cap T$ is a demand cut; i.e., $|\Gamma_T(X \cap T)| = k-1$. Notice that $\Gamma_T(X \cap T) \subseteq \Gamma_T(X) \subseteq \Gamma_{T \cup S}(X)$. Hence, $|\Gamma_T(X \cap T)| = k-1 \geq |\Gamma_{T \cup S}(X)|$ indicates that $\Gamma_T(X \cap T) = \Gamma_T(X) = \Gamma_{T \cup S}(X)$. This implies that $G[S \cup T]$ has no T -path that covers $X \cap T$.

From these observations, we can consider the following simple algorithm for the augmentation problem. Initialize a solution S to an empty set. If $T \cup S$ is not k -connected, there exists a demand cut X that is not covered by any T -path in $G[T \cup S]$. The algorithm chooses such a demand cut X , and adds to S the inner nodes of a T -path covering X that is guaranteed by Lemma 3. The algorithm repeats this until $T \cup S$ becomes k -connected. In fact, this is exactly the same as the algorithms proposed in [17, 21] for $k \leq 3$. Every iteration of this algorithm adds at most two nodes to S . Hence, in order to obtain an approximation

guarantee for this algorithm, it is critical to analyze how many iterations are required to ensure that $T \cup S$ is k -connected.

We analyze the number of iterations for a general connectivity requirement k . To do this, we make a slight modification to the algorithm. We restrict the demand cut X that is chosen in each iteration, as follows. Instead of an arbitrary demand cut X that is not covered by any T -path in $G[T \cup S]$, our algorithm always chooses a minimal of such cuts. This procedure is described in detail as Algorithm 1.

Algorithm 1 Simple Algorithm

Input: an undirected graph $G = (V, E)$ and a $(k - 1, m)$ -CDS $T \subseteq V$ with $m \geq k$

Output: $S \subseteq V \setminus T$ such that $G[T \cup S]$ is k -connected

$S \leftarrow \emptyset$

while $G[T \cup S]$ is not k -connected **do**

$X \leftarrow$ a minimal demand cut that is not covered by any T -path in $G[T \cup S]$

$P \leftarrow$ a minimum-length T -path that covers X

 add the inner nodes in P to S

end while

output S

In the following theorem, we show that $O(k|T|)$ iterations are sufficient to ensure that Algorithm 1 computes a feasible solution.

Theorem 1. *Algorithm 1 outputs a solution after $k(2|T| - 3)$ iterations.*

Each iteration of Algorithm 1 adds at most two nodes to the solution. Hence, Theorem 1 immediately implies that Algorithm 1 outputs a solution S such that $|S| \leq 2k(2|T| - 3)$. Let OPT denote the minimum size of (k, m) -CDSs. If $|T| \leq \alpha \text{OPT}$, then $|T \cup S| \leq 5k\alpha \text{OPT}$. Hence, Theorem 1 indicates that Algorithm 1 achieves an approximation factor of $5k\alpha$ for the unweighted augmentation problem, if T is computed by an α -approximation algorithm for the unweighted $(k - 1, m)$ -CDS problem. Therefore, the following result is obtained.

Corollary 1. *There exists an $O(5^k k!)$ -approximation algorithm for the unweighted (k, m) -CDS problem in unit disk graphs if $m \geq k$.*

We note that our algorithm for the augmentation problem does not rely on any property specific to unit disk graphs, and so the result in Corollary 1 can be extended to any graph class that admits a constant-approximation algorithm for finding a minimum m -dominating set. One such example is the class of bounded clique- and tree-width graphs [6].

In the remainder of this section, we prove Theorem 1. We begin by selecting an arbitrary node $r \in T$. For $S \subseteq V \setminus T$, let $\mathcal{D}(r, S)$ denote the family of all demand cuts X such that $r \notin X^+$, and X is not covered by any T -path in $G[T \cup S]$.

Lemma 5. *Let $r \in T$ and $S \subseteq V \setminus T$. Let $X, Y \in \mathcal{D}(r, S)$. If X is minimal in $\mathcal{D}(r, S)$, then $X \cap Y = \emptyset$ or $X \subseteq Y$ holds. If X is maximal in $\mathcal{D}(r, S)$, then $X \cap Y = \emptyset$ or $Y \subseteq X$ holds.*

Proof. Suppose that some pair of X and Y violates this claim. Note that we must consider two cases: X is minimal or maximal. In both cases, $X \cap Y \neq \emptyset$. If X is minimal, then $X \not\subseteq Y$, and the minimality of X implies that $Y \subset X$ does not hold. If X is maximal, then $Y \not\subseteq X$, and the maximality of X implies that $X \subset Y$ does not hold. Hence, in both cases, $\emptyset \neq X \cap Y \subset X \subset X \cup Y$ holds. Also, $r \notin (X \cup Y)^+$ follows from $r \notin X^+$ and $r \notin Y^+$, and $r \notin (X \cap Y)^+$ follows, because $(X \cap Y)^+ \subseteq (X \cup Y)^+$. Therefore, both $X \cap Y$ and $X \cup Y$ are (T, r) -cuts.

For each T -path P in $G[T \cup S]$, we add an edge joining two end nodes of P to $G[T]$. Let G' denote the graph with the node set T obtained by this operation. Consider inequality (1), where Γ_T is defined with

respect to the graph G' . The left-hand side of (1) is exactly $2(k-1)$, because $X, Y \in \mathcal{D}(r, S)$, and the right-hand side is at least $2(k-1)$, because $X \cap Y$ and $X \cup Y$ are (T, r) -cuts. Hence, the inequality holds with equality, and $|\Gamma_T(X \cap Y)| = |\Gamma_T(X \cup Y)| = k-1$. This means that both $X \cap Y$ and $X \cup Y$ belong to $\mathcal{D}(S, r)$. This contradicts the minimality or the maximality of X . \square

Define \mathcal{A} as the family of demand cuts chosen in the while loop of Algorithm 1. For $r \in T$, let $\mathcal{A}_r = \{X \in \mathcal{A} : r \notin \Gamma(X)\}$.

Lemma 6. $|\mathcal{A}_r| \leq 2|T| - 3$.

Proof. For each $X \in \mathcal{A}_r$, we let X' denote $T \setminus X^+$ if $r \in X$, and otherwise, we denote it as X . Then X' is a (T, r) -cut for any $X \in \mathcal{A}_r$. We can prove that $\{X' : X \in \mathcal{A}_r\}$ is a laminar family on $T \setminus \{r\}$. The lemma follows from this, because the size of a laminar family on the set of cardinality $|T| - 1$ is at most $2|T| - 3$.

Suppose that there exist $X, Y \in \mathcal{A}_r$ such that $X' \cap Y' \neq \emptyset$, $X' \setminus Y' \neq \emptyset$, and $Y' \setminus X' \neq \emptyset$. We may assume without loss of generality that X is chosen in an earlier iteration than that in which Y is chosen. Let S denote the subset at the beginning of the iteration during which X is chosen. Then both X' and Y' belong to $\mathcal{D}(r, S)$. If $r \notin X$, then X' is minimal in $\mathcal{D}(r, S)$, and Lemma 5 shows that $X' \cap Y' = \emptyset$ or $X' \subseteq Y'$. If $r \in X$, then X' is maximal in $\mathcal{D}(r, S)$, and Lemma 5 shows that $X' \cap Y' = \emptyset$ or $Y' \subseteq X'$. In either case, this contradicts the definitions of X and Y . \square

If $X \in \mathcal{A}$ does not belong to \mathcal{A}_r for some $r \in T$, then r is contained in $\Gamma_T(X)$. Recall that $|\Gamma_T(X)| = k-1$ for all demand cuts X . Hence, for any distinct nodes $r_1, \dots, r_k \in T$, we have $\mathcal{A} = \bigcup_{i=1}^k \mathcal{A}_{r_i}$, and so $|\mathcal{A}| \leq \sum_{i=1}^k |\mathcal{A}_{r_i}| \leq k(2|T| - 3)$, where the last inequality follows from Lemma 6. Since Algorithm 1 iterates $|\mathcal{A}|$ times, Theorem 1 has been proven.

5 Primal-dual algorithm for the weighted problem

In this section, we consider the weighted (k, m) -CDS problem. Our algorithm for this problem is also based on a subroutine that solves the augmentation problem. We show that there is a constant-approximation algorithm for the augmentation problem with general node weights. This algorithm is based on the primal-dual method, which is a technique for computing an approximate solution from a linear programming (LP) relaxation of the problem. Before introducing the primal-dual algorithm, we consider the weighted m -dominating set problem, which demands a minimum weight m -dominating set; we prove that the problem admits a constant-approximation algorithm.

5.1 Approximation algorithm for the weighted m -dominating set problem

Our algorithm reduces the weighted m -dominating set problem to the following geometric problem.

Definition 2 (Disk multicover problem). *We are given a set P of points and a set D of disks on the Euclidean plane, a demand $d(p)$ for each point $p \in P$, and a nonnegative weight $w(i)$ for each disk $i \in D$. A subset D' of D is called a disk cover if each point $p \in P$ is contained in at least $d(p)$ disks in D' . The problem requires finding a disk cover D' that minimizes the weight $\sum_{i \in D'} w(i)$.*

When $d(p) = 1$ for all $p \in P$, this is called the *disk cover problem*. We write $p \in i$ if a point p is included in a disk i .

Bansal and Pruhs [3] presented a constant-approximation algorithm for the disk multicover problem. Their algorithm is an LP-rounding algorithm. That is to say, their algorithm first solves the following LP relaxation of the problem:

$$\begin{aligned} & \text{minimize} && \sum_{i \in D} w(i)x(i) \\ & \text{subject to} && \sum_{i \in D: p \in i} x(i) \geq d(p), \quad \forall p \in P, \\ & && 0 \leq x(i) \leq 1, \quad \forall i \in D, \end{aligned} \tag{3}$$

then it computes a disk cover D' that satisfies $\sum_{i \in D'} w(i) = O(1) \cdot \sum_{i \in D} w(i)x(i)$ for an optimal solution x to (3).

When $m = 1$, the problem of finding a minimum weight m -dominating set in a unit disk graph can be reduced to the disk cover problem, as follows. Define D as the set of unit disks corresponding to the nodes in the unit disk graphs, and define P as the set of the centers of the disks. The weight $w(i)$ of a disk $i \in D$ is defined as the weight of the corresponding node in the graph. For each point p , a disk cover in this instance includes at least one disk that contains p . This means that the set of nodes corresponding to the disks in the disk cover is a 1-dominating set of the graph.

From the weighted m -dominating set problem with $m \geq 2$, we can similarly define an instance of the disk multicover problem; D , P , and w are defined in the same way, and the demand $d(i)$ of each disk $i \in D$ is defined as m . By solving this instance, we can obtain an m -dominating set in the unit disk graph. However, the minimum weight of disk covers in the obtained instance is possibly too large, compared with the minimum weight of the m -dominating sets. To see this, let D' be a disk cover in the obtained instance of the disk multicover problem. The constraint in the disk multicover problem demands that each point $i \in D'$ is included in at least $d(i)$ disks in D' . On the other hand, in the weighted m -dominating set problem, if a solution includes a node i , it is feasible even if it does not contain $d(i)$ neighbors of i . In other words, the constraint of the disk multicover problem in the constructed instance is stronger than the one demanded in the original instance of the weighted m -dominating set problem. Accordingly, there does not seem to exist a straightforward reduction from the weighted m -dominating set problem to the disk multicover problem.

Nevertheless, we show that the weighted m -dominating set problem in a unit disk graph can be approximated via an algorithm for the disk multicover problem. Our algorithm first solves the following LP relaxation of the weighted m -dominating set problem:

$$\begin{aligned} & \text{minimize} && \sum_{v \in V} w(v)x(v) \\ & \text{subject to} && \sum_{u \in \Gamma(v) \setminus S} x(u) \geq (m - |S|)(1 - x(v)), \quad \forall v \in V, \forall S \subseteq \Gamma(v), \\ & && 0 \leq x(v) \leq 1, \quad \forall v \in V. \end{aligned} \tag{4}$$

Although (4) has an exponential number of constraints, the separation can be done in polynomial time. Namely, given x , we can judge whether x is a feasible solution for (4), as follows. For each v , sort the neighbors $u_1, \dots, u_i \in \Gamma(v)$ so that $x(u_1) \leq \dots \leq x(u_i)$. If $\sum_{j=1}^{i-m'} x(u_j) \geq (m - m')(1 - x(v))$ for each $m' = 0, \dots, m$, then the constraint defined from v and all $S \subseteq \Gamma(v)$ is satisfied by x . If $\sum_{j=1}^{i-m'} x(u_j) \geq (m - m')(1 - x(v))$ does not hold for some $m' = 0, \dots, m$, then the constraint defined from v and $S := \{u_{i-m'+1}, \dots, u_i\}$ ($S := \emptyset$ when $m' = 0$) is violated by x . Hence, the separation can be done by checking whether the $m+1$ inequalities hold for each node v . Therefore, the ellipsoid method can be used to solve (4) in polynomial time.

Let x^* denote an optimal solution for (4). We define U as $\{v \in V : x^*(v) \leq 1/2\}$. Our algorithm invokes an algorithm for the disk multicover problem after the input is set as follows. D is defined as the set of disks corresponding to the nodes in U , and P is defined as the set of the center of the disks in D . The demand $d(p)$ of the point $p \in P$ corresponding to a node $u \in U$ is defined as $m - |\Gamma(u) \setminus U|$. We solve the obtained instance of the disk multicover problem by using an LP rounding algorithm based on (3), such as the algorithm of Bansal and Pruhs [3]. Let D' be the set of nodes corresponding to the disks in the obtained approximate solution. Our algorithm outputs $D' \cup (V \setminus U)$ as an approximate solution for the weighted m -dominating set problem.

Theorem 2. *Our algorithm approximates the weighted m -dominating set problem in a unit disk graph within a constant factor, if the algorithm for the disk multicover problem computes a solution of weight at most $O(1) \cdot \sum_{i \in D} w(i)x(i)$ for an optimal solution x to (3).*

Proof. Let OPT denote the minimum weight of the m -dominating sets, and let x^* denote an optimal solution for (4). Then, $\sum_{v \in V} w(v)x^*(v) \leq \text{OPT}$ holds because (4) relaxes the weighted m -dominating set problem (i.e., the incidence vector of each m -dominating set is a feasible solution to (4)).

For each $v \in U$, let p_v and i_v respectively denote the point in P and the disk in D corresponding to v . Define $\bar{x} \in [0, 1]^D$ by $\bar{x}(i_v) = 2x^*(v)$ for each $v \in U$. Then, for each $v \in U$,

$$\sum_{i \in D: p_v \in i} \bar{x}(i) = 2 \sum_{u \in \Gamma_U(v) \cup \{v\}} x^*(u) \geq 2(m - |\Gamma(v) \setminus U|)(1 - x^*(v)) \geq m - |\Gamma(v) \setminus U| = d(p_v)$$

holds, where the first inequality follows from the constraint of (4), and the second inequality follows from $x^*(v) \leq 1/2$. Hence, \bar{x} is a feasible solution to (3). The algorithm for the disk multicover problem computes a solution $D' \subseteq D$ such that $\sum_{i \in D'} w(i) \leq \alpha \sum_{i \in D} w(i) \bar{x}(i) \leq 2\alpha \sum_{v \in U} w(v) x^*(v)$ for some constant α . On the other hand, $\sum_{i \in V \setminus U} w(i) < 2 \sum_{v \in V \setminus U} w(v) x^*(v)$, because $x^*(v) > 1/2$ for each $v \in V \setminus U$. Therefore, $\sum_{v \in D' \cup (V \setminus U)} w(v) \leq 2\alpha \sum_{v \in V} w^*(v) x^*(v) \leq 2\alpha \text{OPT}$.

Let u be a node that is not contained in $D' \cup (V \setminus U)$. Then, u has $d(p_u) = m - |\Gamma(u) \setminus U|$ neighbors in D' and $|\Gamma(u) \setminus U|$ neighbors in $(V \setminus U)$. Therefore, $D' \cup (V \setminus U)$ is a required disk cover. \square

A drawback to our algorithm is that it requires the ellipsoid method, which tends to be slow in practice. When $m \in \{k, k+1\}$, this can be avoided by using the above-mentioned straightforward reduction to the disk multicover problem. Recall that the reduction does not work in general because a node in an m -dominating set S may not have m neighbors in S . However, when S is k -connected for some $k \geq m - 1$, each node $v \in S$ has $m - 1$ neighbors in S . Note that v is not counted as a neighbor of v . Hence, the minimum weight of the disk covers can be bounded by the minimum weight of the (k, m) -CDSs, and the straightforward reduction gives an m -dominating set that has a weight within a constant factor of the minimum weight of the (k, m) -CDSs.

5.2 Algorithm for the augmentation problem

First, let us give an overview of our algorithm for the augmentation problem. When the connectivity requirement k is equal to one, the augmentation problem is known as the *node-weighted Steiner tree problem*. For general graphs, it is hard to approximate this problem within a factor of $o(\log n)$, because it extends the set cover problem. However, in unit disk graphs, there is a constant-approximation algorithm for the node-weighted Steiner tree problem. Zou et al. [24] proved the existence of such an algorithm from the fact that any unit disk graph has a spanning tree of maximum degree five. This property of unit disk graphs is well known, and it can be shown by using the following observation: if there is a node v of degree more than five in a spanning tree, then by Lemma 1, there is an edge uu' that joins two neighbors u and u' of v . Replacing the edge vu by another edge uu' transforms the spanning tree into another spanning tree in which the degree of v is decreased by one (to ensure the existence of a spanning tree of maximum degree five, we must consider the degree of u' , because it is increased by the operation). This approach cannot be directly extended to the general connectivity requirement k , because this operation does not preserve the connectivity of a graph. To see this, consider the graph on seven nodes u, v_1, \dots, v_6 such that v_1, \dots, v_6 form a cycle of length six, and u is adjacent to each of v_1, \dots, v_6 . This graph is 3-connected, and the degree of u is six. To decrease the degree of u , replace one edge uv_i by another edge $v_{i-1}v_i$, and then v_i will have only two neighbors; hence, the connectivity of the graph has been decreased to two.

Nevertheless, we will show that Lemma 1 can be used to show that the augmentation problem admits a better approximation algorithm for unit disk graphs than it does for general graphs. We will use the lemma in the framework of primal-dual method, which has been applied successfully to many network design problems [12]. Our algorithm repeats growing several dual variables simultaneously in an LP relaxation. This approach has been used in the augmentation problem with node weights [4, 10], but its approximation factor depends on the number of nodes. This is because the approximation factor is decided by the number of dual variables that are grown simultaneously in a single constraint. In our LP relaxation of the augmentation problem, each dual variable corresponds to a demand cut, and each constraint corresponds to a node in the given graph. Since this number cannot be bounded, the primal-dual method does not achieve a good

approximation factor for general graphs, but we will show that this number can be bounded in unit disk graphs, due to Lemma 1.

Let us explain the detail of our algorithm. We choose a root node $r \in T$, and we consider the problem of finding a minimum weight node set $S \subseteq V \setminus T$ such that every (T, r) -cut X with $|\Gamma_T(X)| = k - 1$ is covered by a T -path in $G[T \cup S]$. By repeating this for k roots, we obtain an approximate solution for the augmentation problem.

For the remainder of this subsection, we fix a root node $r \in T$. We say a Steiner (T, r) -cut X is a *demand cut* if $|\Gamma_T(X)| = k - 1$ (note that this is slightly different from the definition in Section 4). We denote by \mathcal{D} the family of all demand cuts. Observe that S is a feasible solution for the current problem defined from r if and only if $\Gamma(X) \cap S \neq \emptyset$ for each demand cut X . Thus, an LP relaxation of the problem can be formulated as follows:

$$\begin{aligned} & \text{minimize} && \sum_{v \in V \setminus T} w(v)x(v) \\ & \text{subject to} && \sum_{v \in \Gamma(X) \setminus T} x(v) \geq 1, \quad \forall X \in \mathcal{D}, \\ & && x(v) \geq 0, \quad \forall v \in V \setminus T. \end{aligned} \tag{5}$$

The dual of this LP is

$$\begin{aligned} & \text{maximize} && \sum_{X \in \mathcal{D}} y(X) \\ & \text{subject to} && \sum_{X \in \mathcal{D}: v \in \Gamma(X)} y(X) \leq w(v), \quad \forall v \in V \setminus T, \\ & && y(X) \geq 0, \quad \forall X \in \mathcal{D}. \end{aligned} \tag{6}$$

We say that a node $v \in V \setminus T$ *covers* a demand cut X if $v \in \Gamma(X)$, and a node set S covers X if there exists a node $v \in S$ that covers X .

A subfamily \mathcal{F} of \mathcal{D} is called *uncrossable* when (i) any $X, Y \in \mathcal{F}$ satisfy $X \cap Y, X \cup Y \in \mathcal{F}$ or $X \setminus Y^+, Y \setminus X^+ \in \mathcal{F}$, and (ii) if $X \cap Y \cap T \neq \emptyset$, then $X \cap Y, X \cup Y \in \mathcal{F}$ holds. In general, the family \mathcal{D} is not uncrossable. We will present a constant-approximation algorithm for the problem when \mathcal{D} is uncrossable. If \mathcal{D} is not uncrossable, our algorithm finds an uncrossable subfamily \mathcal{F} of \mathcal{D} , and it uses the algorithm for an uncrossable family to find a node set that covers all demand cuts in \mathcal{F} . After adding to the solution all the nodes in the obtained node set, the algorithm updates \mathcal{D} , setting it equal to the residual family, which consists of all the demand cuts that are not covered by the current solution. This is repeated until \mathcal{D} becomes uncrossable. We can prove that the algorithm for an uncrossable family is invoked $O(k)$ times. Indeed, this part of the algorithm (given in Section 5.2.1) is a straightforward application of the result presented by Nutov [16], who used the decomposition into uncrossable subfamilies to design an algorithm for a node-connectivity survivable network design problem.

Below, we first explain how to find an uncrossable subfamily of \mathcal{D} when \mathcal{D} is not uncrossable in Section 5.2.1, and then we present an algorithm for covering an uncrossable family in Section 5.2.2.

5.2.1 Decomposition into uncrossable subfamilies

We note that the result here can be found in Nutov [16]. We present detailed proofs of all results in this subsection for completeness.

Let a *min-core* signify a minimal demand cut in \mathcal{D} , and let \mathcal{M} be the family of all min-cores. If a demand cut X includes only one min-core as a subset (i.e., $|\{Y \in \mathcal{M}: Y \subseteq X\}| = 1$), then it is called a *core*. The core family of $X \in \mathcal{M}$, denoted by $\mathcal{C}(X)$, is the family of cores that include X .

We first characterize the case when \mathcal{D} and a family of cores are uncrossable.

Lemma 7.

- The core family of $Z \in \mathcal{M}$ is a ring family; i.e., for any $X, Y \in \mathcal{C}(Z)$, $X \cap Y, X \cup Y \in \mathcal{C}(Z)$ holds.
- Let $Z, W \in \mathcal{M}$ be distinct min-cores. If $X \in \mathcal{C}(Z)$ and $Y \in \mathcal{C}(W)$, then $X \cap Y \cap T = \emptyset$.
- The family \mathcal{D} of all demand cuts is uncrossable if and only if there are no $X, Y \in \mathcal{D}$ such that $X \cap T \subseteq \Gamma(Y)$.

- A family $\mathcal{C}' := \bigcup_{Z \in \mathcal{M}'} \mathcal{C}(Z)$ defined from $\mathcal{M}' \subseteq \mathcal{M}$ is uncrossable if and only if there are no $X, Y \in \mathcal{C}'$ such that $X \cap T \subseteq \Gamma(Y)$.

Proof. Let $X, Y \in \mathcal{D}$. If

$$X \cap Y \cap T \neq \emptyset \quad (7)$$

holds, both $X \cap Y$ and $X \cup Y$ are Steiner (T, r) -cuts. In this case, the right-hand side of (1) is at least $2(k-1)$. Since the left-hand side of (1) is equal to $2(k-1)$, the inequality holds with equality, which implies $X \cap Y, X \cup Y \in \mathcal{D}$.

If $X, Y \in \mathcal{C}(Z)$ for some $Z \in \mathcal{M}$, (7) always holds, because $\emptyset \neq Z \cap T \subseteq X \cap Y \cap T$. Hence, $\mathcal{C}(Z)$ is a ring family. Let $X \in \mathcal{C}(Z)$ and $Y \in \mathcal{C}(W)$ for some $Z, W \in \mathcal{M}$ with $Z \neq W$. If (7) holds, $X \cap Y$ includes a min-core as a subset. However, this implies that X or Y includes two different min-cores as subsets, which contradicts the fact that X and Y are cores. Therefore, the first two conditions are proven.

Next, let us prove the remaining two conditions. Let $X, Y \in \mathcal{D}$. If $X \cap T \subseteq \Gamma(Y)$, then neither $X \cap Y$ nor $X \setminus Y^+$ belongs to \mathcal{D} , because $X \cap Y$ and $X \setminus Y^+$ contain no node in T . Hence, \mathcal{D} is not uncrossable in this case. If $X, Y \in \mathcal{C}'$ and $X \cap T \subseteq \Gamma(Y)$, for the same reason as in the previous case, \mathcal{C}' is not uncrossable.

For the remainder of the proof, we will assume that $X \cap T \not\subseteq \Gamma(Y)$ and $Y \cap T \not\subseteq \Gamma(X)$. In this case, (7) or

$$(X \cap T) \setminus Y^+ \neq \emptyset \neq (Y \cap T) \setminus X^+ \quad (8)$$

holds. As observed above, $X \cap Y, X \cup Y \in \mathcal{D}$ holds when (7) holds. If (8) holds, both $X \setminus Y^+$ and $Y \setminus X^+$ are Steiner (T, r) -cuts. Hence, as in (7), (2) holds with equality, implying $X \setminus Y^+, Y \setminus X^+ \in \mathcal{D}$. Therefore, \mathcal{D} is uncrossable.

Let $X, Y \in \mathcal{C}' := \bigcup_{Z \in \mathcal{M}'} \mathcal{C}(Z)$. If $X, Y \in \mathcal{C}(Z)$ for some $Z \in \mathcal{M}'$, we have already seen that $X \cap Y, X \cup Y \in \mathcal{C}'$ holds. Suppose that $X \in \mathcal{C}(Z)$ and $Y \in \mathcal{C}(W)$ for some distinct $Z, W \in \mathcal{M}'$. In this case, as seen above, (7) does not hold, and hence (8) must hold. If $X \setminus Y^+$ does not include Z , then X includes two min-cores. Since this contradicts the definition of X , we have $X \setminus Y^+ \in \mathcal{C}(Z) \subseteq \mathcal{C}'$. With similar reasoning, we can see that $Y \setminus X^+ \in \mathcal{C}(W) \subseteq \mathcal{C}'$. The uncrossability of \mathcal{C}' follows from these relationships. \square

We will use *max-core* to denote a maximal core. We note that a ring family has a unique maximal element, and so $\mathcal{C}(X)$ includes a unique max-core for each $X \in \mathcal{M}$. Let $X, Y \in \mathcal{M}$, and let X' and Y' be the max-cores of $\mathcal{C}(X)$ and $\mathcal{C}(Y)$, respectively. We say that X and Y are *dependent* if $Y \cap T \subseteq \Gamma(X')$, or if $X \cap T \subseteq \Gamma(Y')$. Otherwise, X and Y are called *independent*. If a set of min-cores are pairwise independent, then the union of the core families of those min-cores is uncrossable, as shown in the following lemma.

Lemma 8. *Let $\mathcal{M}' \subseteq \mathcal{M}$. If each pair of $X, Y \in \mathcal{M}'$ is independent, then $\bigcup_{X \in \mathcal{M}'} \mathcal{C}(X)$ is uncrossable.*

Proof. For brevity, let \mathcal{C}' denote $\bigcup_{X \in \mathcal{M}'} \mathcal{C}(X)$. Suppose that \mathcal{C}' is not uncrossable. Then, by Lemma 7, there are two cores $X, Y \in \mathcal{C}'$ such that $X \cap T \subseteq \Gamma(Y)$. Let $X \in \mathcal{C}(Z)$ and $Y \in \mathcal{C}(W)$ for min-cores $Z, W \in \mathcal{M}'$. Then $Z \cap T \subseteq X \cap T \subseteq \Gamma(Y)$. Let W' be the max-core of $\mathcal{C}(W)$. Since $Y \subseteq W'$, each node in $\Gamma(Y) \setminus \Gamma(W')$ is included in W' . By the second condition in Lemma 7, $Z \cap W' \cap T = \emptyset$ holds. Therefore, $Z \cap T \subseteq \Gamma(W')$ holds, which means that Z and W are dependent. Since this contradicts the definition of \mathcal{M}' , the lemma is proven. \square

Let $\gamma := \min_{X \in \mathcal{M}} |X \cap T|$. Note that if $\gamma \geq k$, there is no $X, Y \in \mathcal{D}$ with $X \cap T \subseteq \Gamma(Y)$, i.e., \mathcal{D} is uncrossable. Hence we consider the case of $\gamma \leq k-1$. In this case, we first divide the family of all cores into $2\lfloor (k-1)/\gamma \rfloor + 1$ uncrossable families. This can be done by dividing \mathcal{M} into subfamilies, each of which consists of pairwise independent min-cores. The details are explained in the following lemma.

Lemma 9. *Let k' denote $\lfloor (k-1)/\gamma \rfloor$. There exist $\mathcal{M}_1, \dots, \mathcal{M}_{2k'+1} \subseteq \mathcal{M}$ such that $\bigcup_{i=1}^{2k'+1} \mathcal{M}_i = \mathcal{M}$ and each pair of min-cores in \mathcal{M}_i is independent for all $i = 1, \dots, 2k' + 1$.*

Proof. We consider a digraph whose node set is \mathcal{M} . The digraph contains an arc from a node corresponding to $X \in \mathcal{M}$ to another node corresponding to $Y \in \mathcal{M}$ if $X \cap T \subseteq \Gamma(Y')$, where Y' is the max-core in $\mathcal{C}(Y)$. The node sets in $\{X \cap T : X \in \mathcal{M}\}$ are disjoint, based on the second condition in Lemma 7, and $|\Gamma_T(Y')| = k - 1$ holds for each demand cut Y' . Hence the in-degree of each node in the digraph is at most k' . Let \bar{G} denote the undirected graph obtained from the digraph by ignoring the orientations of the arcs. Any induced subgraph of \bar{G} has a node of degree at most $2k'$. It is known that the chromatic number of such a graph is at most $2k' + 1$. Namely, the node set of \bar{G} can be decomposed into $2k' + 1$ independent sets. If we define \mathcal{M}_i to be the family of min-cores corresponding to the i -th independent set in the decomposition, then each pair of min-cores in \mathcal{M}_i is independent. \square

Our algorithm repeatedly adds nodes to a solution, and when it terminates, the solution covers all the demand cuts. At a given iteration, if the family of all demand cuts uncovered by the current solution is not uncrossable, then the algorithm uses Lemma 9 to decompose the family of cores into $2\lfloor(k-1)/\gamma\rfloor + 1$ uncrossable families. Then, for each of the uncrossable families, the algorithm finds nodes that cover all the demand cuts therein, and those nodes are added to the solution. Let \mathcal{D}_i denote the family of demand cuts that is not covered by the solution when the i -th iteration begins, and let γ_i denote $\min_{X \in \mathcal{D}_i} |X \cap T|$. Notice that $\mathcal{D}_1 = \mathcal{D}$, and $\gamma_{i+1} \geq 2\gamma_i$ holds because each demand cut in \mathcal{D}_{i+1} includes at least two min-cores in \mathcal{D}_i . Hence, the number of uncrossable families constructed in the algorithm is at most $\sum_{i=0}^{\lceil \log(\gamma_1/(k-1)) \rceil} \{(k-1)/(2^i \gamma_1) + 1\} = O(k/\gamma_1)$. Since $\gamma_1 \geq 1$, if we have a constant-approximation algorithm for covering an uncrossable family of demand cuts, we have an $O(k)$ -approximation algorithm for covering all demand cuts.

5.2.2 Covering an uncrossable family of demand cuts

Here, we explain how to cover an uncrossable family \mathcal{F} of demand cuts. First, we introduce several properties of an uncrossable family.

Lemma 10. *Let \mathcal{F} be an uncrossable family of subsets of V . Let $X, Y \in \mathcal{F}$. If X is a min-core of \mathcal{F} , then either $X \subseteq Y$ or $X \cap Y^+ = \emptyset = X^+ \cap Y$ holds. In particular, if both X and Y are min-cores of \mathcal{F} , the latter condition holds.*

Proof. We note that $X \cap Y^+ = \emptyset = X^+ \cap Y$ holds if and only if $X \cap Y^+ = \emptyset$. Suppose that $X \not\subseteq Y$ and $X \cap Y^+ \neq \emptyset$. Since \mathcal{F} is uncrossable, $X \cap Y, X \cup Y \in \mathcal{F}$ or $X \setminus Y^+, Y \setminus X^+ \in \mathcal{F}$ holds. Note that $X \not\subseteq Y$ implies $X \cap Y \subset X$, and $X \cap Y^+ \neq \emptyset$ implies $X \setminus Y^+ \subset X$. Hence, if either $X \cap Y \in \mathcal{F}$ or $X \setminus Y^+ \in \mathcal{F}$ holds, we have a contradiction with the minimality of X . \square

For $S \subseteq V$, let \mathcal{F}_S denote $\{X \in \mathcal{F} : S \cap \Gamma(X) = \emptyset\}$.

Lemma 11. *If $\mathcal{F} \subseteq 2^V$ is uncrossable, then \mathcal{F}_S is also uncrossable for any $S \subseteq V$.*

Proof. Let $X, Y \in \mathcal{F}_S$. Since \mathcal{F} is uncrossable, $X \cap Y, X \cup Y \in \mathcal{F}$ or $X \setminus Y^+, Y \setminus X^+ \in \mathcal{F}$ holds.

Suppose that the former holds. If $X \cap Y \notin \mathcal{F}_S$, then $\Gamma(X \cap Y)$ includes a node $v \in S$. Since $\Gamma(X \cap Y) \subseteq \Gamma(X) \cup \Gamma(Y)$, v covers X or Y . However, this contradicts the fact that $X, Y \in \mathcal{F}_S$. Hence, $X \cap Y \in \mathcal{F}_S$. Similarly, we can prove $X \cup Y \in \mathcal{F}_S$, because $\Gamma(X \cup Y) \subseteq \Gamma(X) \cup \Gamma(Y)$.

Next, suppose that the latter holds. We note that $\Gamma(X \setminus Y^+) \subseteq \Gamma(X) \cup \Gamma(Y)$ and $\Gamma(Y \setminus X^+) \subseteq \Gamma(X) \cup \Gamma(Y)$. Hence, as above, we can prove that $X \setminus Y^+, Y \setminus X^+ \in \mathcal{F}_S$. \square

Now we present our algorithm for covering an uncrossable family \mathcal{F} of demand cuts. The algorithm consists of an increase phase and a deletion phase. First, we present the increase phase. In this phase, the algorithm maintains the following variables.

- The dual solution y . This is initialized as $y(X) := 0$, $X \in \mathcal{F}$. During the increase phase, y is always a feasible solution to (6).

- A solution $S \subseteq V \setminus T$. This is initialized to $S := \emptyset$. The increase phase terminates when S covers all the demand cuts in \mathcal{F} .

In each iteration, the algorithm simultaneously increases $y(X)$ for each min-core X of \mathcal{F}_S . For ease of presentation, we will consider this over time. During ϵ units of time, $y(X)$ is increased by ϵ for each min-core X of \mathcal{F}_S . When $\sum_{X \in \mathcal{F}: v \in \Gamma(X)} y(X)$ becomes equal to $w(v)$ for some $v \in V \setminus (T \cup S)$, the algorithm stops increasing y and adds v to S . After this update, if S covers all demand cuts in \mathcal{F} , then the algorithm terminates the increase phase and proceeds to the deletion phase. Otherwise, the algorithm proceeds to the next iteration of the increase phase. By definition, in these steps, y is always a feasible solution to (6). At the end of the increase phase, S covers all the demand cuts in \mathcal{F} .

Suppose that the increase phase starts at time 0 and ends at time Δ . Let $\tau \in [0, \Delta]$ be an instant during the increase phase. Let $S_\tau, \mathcal{F}_\tau, \mathcal{M}_\tau$ denote S, \mathcal{F}_S , and the family of min-cores of \mathcal{F}_S , respectively, at time τ . For each $v \in V$, let $d_\tau(v)$ denote $|\{X \in \mathcal{M}_\tau: v \in \Gamma(X)\}|$. Let \mathcal{L} be the family of all demand cuts X that were in \mathcal{M}_τ at time τ . For the analysis given below, we observe the following properties.

Lemma 12. \mathcal{L} is strongly laminar.

Proof. Let $X, Y \in \mathcal{L}$. Suppose that $X \in \mathcal{M}_\tau$ and $Y \in \mathcal{M}_{\tau'}$ for some $\tau \leq \tau'$. Then, $Y \in \mathcal{F}_\tau$, because $Y \in \mathcal{M}_{\tau'}$ means that Y is not covered by S_τ . Hence, by Lemma 10, $X \subseteq Y$ or $X \cap Y^+ = \emptyset = X^+ \cap Y$ holds. \square

Lemma 13. If G is a unit disk graph, $d_\tau(v) \leq 5$ holds for any $v \in V \setminus T$ and $\tau \in [0, \Delta]$.

Proof. Let $X_1, \dots, X_{d_\tau(v)}$ be the members of \mathcal{M}_τ whose neighbor sets include v . Let u_i denote a neighbor of v in X_i for each $i = 1, \dots, d_\tau(v)$. Notice that by Lemma 11, \mathcal{F}_τ is uncrossable. Thus, if $i \neq j$, then by Lemma 10, $X_i \cap X_j^+ = \emptyset = X_i^+ \cap X_j$. If $d_\tau(v) \geq 6$, u_i and u_j are adjacent for some $i, j \in \{1, \dots, d_\tau(v)\}$ with $i \neq j$. However, this indicates that $u_j \in X_i^+$, which contradicts $X_j \cap X_i^+ = \emptyset$. \square

In the deletion phase, the algorithm modifies S into an inclusionwise minimal node set that covers \mathcal{F} as follows. Let $S := \{v_1, \dots, v_{|S|}\}$, where v_i is the i -th node added to S in the increase phase for each $i = 1, \dots, |S|$. The deletion phase investigates nodes $v_i \in S$ in decreasing order of their subscripts. If $S \setminus \{v_i\}$ covers all demand cuts in \mathcal{F} , v_i is removed from S . Let \tilde{S} denote S after all nodes have been investigated. The algorithm outputs \tilde{S} as a solution. We will show that \tilde{S} is a 15-approximate solution.

Lemma 14. Let $\tau \in [0, \Delta]$, and let $v_i \in \tilde{S}$ with $d_\tau(v_i) \geq 1$. There exist $W \in \mathcal{F}_\tau$ and $X \in \mathcal{M}_\tau$ such that $\Gamma(W) \cap (\tilde{S} \cup \{v_1, \dots, v_i\}) = \{v_i\}$, $v_i \in \Gamma(X)$, and $X \subseteq W$ or $X \cap W^+ = \emptyset = X^+ \cap W$.

Proof. By the definition of the deletion phase, $v_i \in \tilde{S}$ implies that there exists $W \in \mathcal{F}$ such that $\Gamma(W) \cap (\tilde{S} \cup \{v_1, \dots, v_i\}) = \{v_i\}$. Note that $W \in \mathcal{F}_\tau$ holds because no node in $\{v_1, \dots, v_{i-1}\}$ covers W . Since $d_\tau(v_i) \geq 1$, there exists $X \in \mathcal{M}_\tau$ with $v_i \in \Gamma(X)$. By Lemma 10, $X \subseteq W$ or $X \cap W^+ = \emptyset = X^+ \cap W$ holds. \square

For $v_i \in \tilde{S}$, we will call (W, X) in Lemma 14 a *witness pair* of v_i .

Lemma 15. $\sum_{v \in \tilde{S}} d_\tau(v) \leq 15|\mathcal{M}_\tau| - 5$ for any $\tau \in [0, \Delta]$.

Proof. Let v be a node in \tilde{S} such that $d_\tau(v) \geq 1$, from which $v \notin S_\tau$ follows by Lemma 13. We categorize such a node v into two types. If there exists a witness pair (W_v, X_v) of v such that $X_v \subseteq W_v$, v is said to be of the first type; otherwise, v is said to be of the second type.

Let us count the number of nodes v of the first type. Let $X \in \mathcal{M}_\tau$. Suppose that there are two nodes of the first type, $u, v \in \tilde{S}$, such that $X = X_u = X_v$. Then $X \subseteq W_u \cap W_v$ holds. Recall that $\Gamma(W_u) \cap \tilde{S} = \{u\}$, $\Gamma(W_v) \cap \tilde{S} = \{v\}$, and $u, v \in \Gamma(X) \subseteq (W_u \cap W_v)^+$. These imply that $\Gamma(W_u \cup W_v) \cap \tilde{S} = \emptyset$. Since $W_u \cap W_v \cap T \supseteq X \cap T \neq \emptyset$, the definition of uncrossability indicates that $W_u \cup W_v \in \mathcal{F}$. However, this contradicts the definition of \tilde{S} . Therefore, for each $X \in \mathcal{M}_\tau$, there exists at most one node of the first type with $X_v = X$. That is to say, there are at most $|\mathcal{M}_\tau|$ nodes of the first type.

Next, we count the number of nodes of the second type. Let v be a node of the second type. There exists $Y \in \mathcal{M}_\tau$ such that $Y \subseteq W_v$, because W_v is not covered by any node that was added to S earlier than v in the increase phase. Since v is not the node of the first type, v is not included in $\Gamma(Y)$, and thus $Y \neq X_v$. Let $\mathcal{L}_\tau = \bigcup_{\tau' \in [\tau, \Delta]} \mathcal{M}_{\tau'}$. Since $\mathcal{L}_\tau \subseteq \mathcal{L}$, and by Lemma 12, \mathcal{L} is a strongly laminar family, \mathcal{L}_τ is a strongly laminar family. We assume that \mathcal{L}_τ has a unique maximal member; if there is more than one maximal member of \mathcal{L}_τ , we add a node set V to \mathcal{L}_τ , which has no effect on the following discussion. Let Z be a member of \mathcal{L}_τ such that Z became a min-core of the residual family of \mathcal{F} when v was added to S in the increase phase. Then $v \in Z$, and hence $X_v \subseteq Z$ and $Y \subseteq W_v \subseteq Z$, because $v \in X_v^+$ and $v \in W_v^+$. Let Z_v be a minimal member of \mathcal{L}_τ such that $X_v \subseteq Z_v$ and $W_v \subseteq Z_v$. Note that Z_v has at least two children in \mathcal{L}_τ . Suppose that there exists a node of the second type, $u \in \tilde{S} \setminus \{v\}$, such that $d_\tau(u) \geq 1$ and $W_v \subseteq W_u \subset Z_v$. Then, $v \in W_u$ holds, because $\Gamma(W_u) \cap \tilde{S} = \{u\}$, and $X_v \subseteq W_u$ because of the strong laminarity of \mathcal{L}_τ . Since this contradicts the definition of Z_v , no such u exists. In summary, this means that the number of nodes of the second type is at most $\sum_{Z \in \mathcal{L}_\tau: \text{ch}(Z) \geq 2} \text{ch}(Z)$, where $\text{ch}(Z)$ denotes the number of children of Z in \mathcal{L}_τ . Since the leaf set of \mathcal{L}_τ is \mathcal{M}_τ , we have $\sum_{Z \in \mathcal{L}_\tau: \text{ch}(Z) \geq 2} \text{ch}(Z) \leq 2|\mathcal{M}_\tau| - 1$.

Thus, $|\{v \in \tilde{S}: d_\tau(v) \geq 1\}| \leq 3|\mathcal{M}_\tau| - 1$, and $d_\tau(v) \leq 5$ for each $v \in \tilde{S}$, and the lemma has been proven. \square

Theorem 3. *If \mathcal{F} is an uncrossable family of demand cuts, there exists a 15-approximation algorithm for finding a minimum weight node set that covers all demand cuts in \mathcal{F} .*

Proof. We will prove that the algorithm presented above is a 15-approximation algorithm. By definition, the algorithm computes a feasible solution \tilde{S} to the problem, and a solution $y \in \mathbb{R}_+^{\mathcal{F}}$ is a feasible solution to (6). Since $\sum_{X \in \mathcal{F}} y(X)$ is a lower bound on the optimal value, it suffices to prove that $\sum_{v \in \tilde{S}} w(v) \leq 15 \sum_{X \in \mathcal{F}} y(X)$.

When the increase phase terminates, y satisfies $\sum_{X \in \mathcal{F}} y(X) = \int_0^\Delta |\mathcal{M}_\tau| d\tau$. Moreover, for each node $v \in V \setminus T$ and $\tau \in [0, \Delta]$, $\frac{d}{d\tau} \sum_{X \in \mathcal{F}: v \in \Gamma(X)} y(X) = d_\tau(v)$ holds. For each $v \in \tilde{S}$, $w(v) = \int_0^\Delta d_\tau(v) d\tau$ holds, because $w(v) = \sum_{X \in \mathcal{F}: v \in \Gamma(X)} y(X)$ holds when the algorithm terminates. By Lemma 15, $\sum_{v \in \tilde{S}} w(v) = \sum_{v \in \tilde{S}} \int_0^\Delta d_\tau(v) d\tau \leq 15 \int_0^\Delta |\mathcal{M}_\tau| d\tau = 15 \sum_{X \in \mathcal{F}} y(X)$. \square

Although we illustrated the algorithm by using a continuous measure of time, it can be easily discretized. Algorithm 2 shows the details of our algorithm for covering an uncrossable family of demand cuts.

5.3 Combined decomposition and covering algorithm

We now summarize our algorithm for the augmentation problem. In Section 5.2.1, we showed that by applying an algorithm for covering an uncrossable family of demand cuts $O(k)$ times, we can find a node set that covers all demand cuts. Since Theorem 3 gives a constant-approximation algorithm for covering an uncrossable family, we have an $O(k)$ -approximation algorithm for covering all demand cuts. Recall that a demand cut is defined as a Steiner (T, r) -cut for a fixed $r \in T$. For covering all Steiner T -cuts X with $|\Gamma_T(X)| = k - 1$, we choose k nodes $r_1, \dots, r_k \in T$, and apply the algorithm fixing r to each of r_1, \dots, r_k . The union of the obtained solutions is an $O(k^2)$ -approximate solution for the augmentation problem. Therefore, we arrive at the following conclusion.

Corollary 2. *The augmentation problem admits an $O(k^2)$ -approximation algorithm. It outputs a solution S such that $\sum_{v \in S} w(v)$ is at most $O(k^2)$ times the optimal value of (5).*

The details of our algorithm for the augmentation problem are given in Algorithm 3.

For the weighted (k, m) -CDS problem, we first compute an m -dominating set T , using the algorithm given in Section 5.1. We incrementally increase the connectivity of T by solving the augmentation problem. This obviously gives an $O(k^3)$ -approximation algorithm. This approximation factor can be slightly improved, as follows.

Algorithm 2 Covering algorithm for an uncrossable family of demand cuts

Input: a unit disk graph $G = (V, E)$, $T \subseteq V$, a nonnegative weight $w(v)$ of each node $v \in V \setminus T$, and an uncrossable family $\mathcal{F} \subseteq \mathcal{D}$

Output: $S \subseteq V \setminus T$ that covers all demand cuts in \mathcal{F}

```
 $S \leftarrow \emptyset, i \leftarrow 0$ 
 $\bar{w}(v) \leftarrow w(v)$  for each  $v \in V \setminus T$ 
while  $\mathcal{F}_S := \{X \in \mathcal{F} : S \cap \Gamma(X) = \emptyset\} \neq \emptyset$  do
   $i \leftarrow i + 1$ 
   $\mathcal{M} \leftarrow \{\text{min-cores of } \mathcal{F}_S\}$ 
   $d(v) \leftarrow |\{X \in \mathcal{M} : v \in \Gamma(X)\}|$  for each  $v \in V \setminus (T \cup S)$ 
   $\alpha \leftarrow \min_{v \in V \setminus (T \cup S)} \bar{w}(v)/d(v)$ 
   $v_i \leftarrow \arg \min_{v \in V \setminus (T \cup S)} \bar{w}(v)/d(v)$ 
   $S \leftarrow S \cup \{v_i\}$ 
   $\bar{w}(v) \leftarrow \bar{w}(v) - \alpha d(v)$  for each  $v \in V \setminus (T \cup S)$ 
end while
for  $j = i - 1, \dots, 1$  do
  if  $S \setminus \{v_j\}$  covers  $\mathcal{F}$ , then  $S \leftarrow S \setminus \{v_j\}$ 
end for
output  $S$ 
```

Algorithm 3 Algorithm for the augmentation problem

Input: Natural numbers k and m with $m \geq k$, a k -connected unit disk graph $G = (V, E)$, an m -dominating set $T \subseteq V$ such that $G[T]$ is $(k - 1)$ -connected, and a nonnegative weight $w(v)$ for each $v \in V \setminus T$

Output: $S \subseteq V \setminus T$ such that $G[T \cup S]$ is k -connected

```
 $S \leftarrow \emptyset$ 
for  $k' = 1, \dots, k$  do
   $r_{k'} \leftarrow T \setminus \{r_1, \dots, r_{k'-1}\}$ 
   $\mathcal{D} \leftarrow \{X \subseteq V : T \cap X \neq \emptyset, r_{k'} \notin X^+, |\Gamma(X) \cap (T \cup S)| = k - 1\}$ 
   $\gamma \leftarrow \min_{X \in \mathcal{D}} |X \cap T|$ 
  while  $\gamma \leq k - 1$  do
     $\mathcal{M} \leftarrow \{\text{min-cores of } \mathcal{D}\}$ 
    for each  $X \in \mathcal{M}$  do  $\mathcal{C}(X) \leftarrow \text{core family of } X \text{ in } \mathcal{D}$ 
    decompose  $\mathcal{M}$  into pairwise independent families  $\mathcal{M}_1, \dots, \mathcal{M}_{\lfloor (k-1)/\gamma \rfloor} \subseteq \mathcal{M}$ 
    for  $i = 1, \dots, \lfloor (k-1)/\gamma \rfloor$  do
      run Algorithm 2 for  $(G, T \cup S, w, \bigcup_{X \in \mathcal{M}_i} \mathcal{C}(X))$  to obtain  $S_i \subseteq V \setminus (T \cup S)$ 
    end for
     $S \leftarrow S \cup \left( \bigcup_{i=1}^{\lfloor (k-1)/\gamma \rfloor} S_i \right)$ 
     $\mathcal{D} \leftarrow \{X \in \mathcal{D} : \Gamma(X) \cap S = \emptyset\}$ 
     $\gamma \leftarrow \min_{X \in \mathcal{D}} |X \cap T|$ 
  end while
  run Algorithm 2 for  $(G, T \cup S, w, \mathcal{D})$  to obtain  $S' \subseteq V \setminus (T \cup S)$ 
   $S \leftarrow S \cup S'$ 
end for
output  $S$ 
```

Corollary 3. *There exists an $O(k^2 \log k)$ -approximation algorithm for the weighted (k, m) -CDS problem.*

Proof. Let S^* denote an optimal solution for the weighted (k, m) -CDS problem, and let $x^* \in \{0, 1\}^V$ denote its characteristic vector (i.e., $x^*(v) = 1$ if $v \in S^*$, and $x^*(v) = 0$ if $v \in V \setminus S^*$). For each $S \subseteq V$, we abbreviate $\sum_{v \in S} w(v)$ to $w(S)$.

We will show that the algorithm described above achieves the approximation factor $O(k^2 \log k)$. Recall that we use the algorithm given in Corollary 2 for the augmentation problem, but with the connectivity requirement k' changed from 1 to k . Let $S_{k'}$ denote the solution output by the algorithm for the augmentation problem when the connectivity requirement is k' . Note that in this case, the node set T in the input is $\bigcup_{i=0}^{k'-1} S_i$, where S_0 is the m -dominating set computed by the algorithm given in Section 5.1. Note that $S^* \cup \left(\bigcup_{i=0}^{k'-1} S_i\right)$ is k -connected. Hence, when $T = \bigcup_{i=0}^{k'-1} S_i$, for each demand cut X , $\Gamma(X) \setminus T$ includes at least $k - k' + 1$ nodes in S^* . This implies that $x^*/(k - k' + 1)$ is a feasible solution to (5) when the connectivity requirement is k' , and thus $w(S_{k'}) = O(k^2) \cdot w(S^*)/(k - k' + 1)$. For the solution output by our algorithm for the weighted (k, m) -CDS problem, the weight is at most

$$\sum_{k'=0}^k w(S_{k'}) = O(k^2) \cdot w(S^*) \sum_{k'=0}^k \frac{1}{k - k' + 1} = O(k^2 \log k) \cdot w(S^*).$$

□

6 Simulation results

We implemented our algorithms and evaluated their performance for computer simulations of the unweighted (k, m) -CDS problem. For computing a connected m -dominating set, we combined the algorithm of Shang et al. [17] with a simple heuristic. The program computes a connected m -dominating set using the algorithm in [17], and then transforms it to form an inclusion-wise minimal connected m -dominating set by using a greedy method to remove nodes from the solution. After this operation, the connectivity of the solution is increased by the algorithm in Section 4 (simple algorithm) or by the one in Section 5 (primal-dual algorithm). We prepared two types of unit disk graphs; nodes were randomly and uniformly generated in a square region with sides of length 100 for the first type and in a 50 by 200 rectangular region for the other. In both types, nodes were joined by an edge if their distance was less than or equal to 20.

Figure 2 shows the sizes of the $(5, 5)$ -CDSs computed by our algorithms. We changed the number of nodes in the graph from 200 to 900. For the solutions computed by the simple algorithm, in the square region, the minimum size was 84 (when $|V| = 600$), and the maximum was 101 (when $|V| = 800$); and in the rectangular region, the minimum size was 97 (when $|V| = 700$), and the maximum was 109 (when $|V| = 900$). For the solutions computed by the primal-dual algorithm, in the square region, the minimum size was 81 (when $|V| = 600$), and the maximum was 95 (when $|V| = 800$); and in the rectangular region, the minimum size was 93 (when $|V| = 600$), and the maximum was 103 (when $|V| = 800$). We conclude that the size of the solution does not increase very much compared with the increase in the size of the graph.

Figure 3 shows the size of the $(k, 6)$ -CDSs and the (k, k) -CDSs that were computed by the two algorithms for $k = 1, \dots, 6$. Here, the graph was on 600 nodes in the rectangular region. We can see that the size of the (k, k) -CDS increases almost linearly as k increases from 1 to 6. When the domination number is fixed at 6, the size of the $(k, 6)$ -CDSs increases slowly. This means that the step in which the $(1, m)$ -CDS is computed has more influence than the step in which the connectivity is increased. The simulation results indicate that the heuristic we used for the construction of the $(1, m)$ -CDS is effective. For example, when the graph is on 600 nodes in a square region, the algorithm of Shang et al. computes a $(1, 6)$ -CDS with 131 nodes, and our heuristic removes 40 of those nodes. From the obtained $(1, 6)$ -CDS with 91 nodes, the simple and primal-dual algorithms computed $(6, 6)$ -CDSs with 116 nodes and 113 nodes, respectively. Therefore, both of our algorithms output $(6, 6)$ -CDSs smaller than the $(1, 6)$ -CDS computed by the algorithm of Shang et al. This can be observed in each of the simulations we performed.

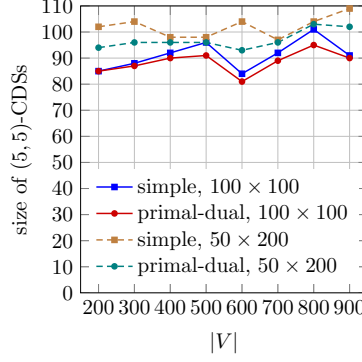


Figure 2: Solution size for various graph sizes

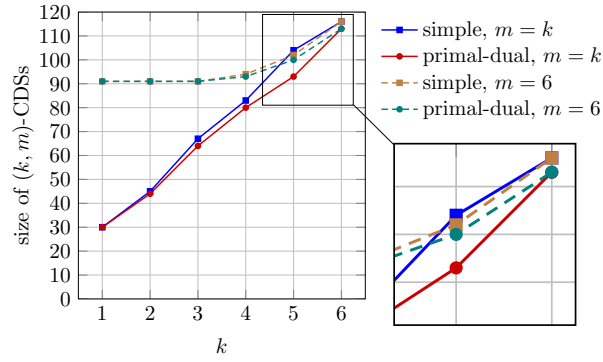


Figure 3: Solution size for various connectivity and domination numbers

In each of the above results, the solutions of the primal-dual algorithm were smaller than those of the simple algorithm. However, in most cases, this difference was extremely small, although their approximation factors were very different.

7 Conclusion

We presented two constant-approximation algorithms for the unweighted (k, m) -CDS problem with $k \geq 4$, and for the weighted (k, m) -CDS problem with $(k, m) \neq (1, 1)$ in unit disk graphs. The first of these is a simple algorithm that can be applied to a fairly general class of graphs, although it is restricted to the unweighted (k, m) -CDS problem. The second is a primal-dual algorithm that has a better approximation factor and can be applied to the weighted (k, m) -CDS problem.

Computational simulations indicated that the performance of the simple algorithm was not much worse than that of the primal-dual algorithm, although the approximation factor of the simple algorithm was exponential in k . This approximation factor may be too high, because our analysis was not specific to unit disk graphs except when computing the m -dominating set. It will be an interesting future work to perform a better analysis of the simple algorithm for unit disk graphs.

In addition to the m -dominating sets, there are many other variations of dominating sets in graphs. For example, a subset S of a node set V is called an m -tuple dominating set if each node in the graph (including those in S) has m neighbors in S , and it is called a vector dominating set if each node v outside of S has $d(v)$ nodes in S for a given $|V|$ -dimensional vector d . Refer to [7] for other variations. Our algorithms for the augmentation problem can be used for increasing the connectivity of these variations if each node outside the solution has k neighbors in the solution, where k is the required connectivity.

For the weighted problem, our primal-dual algorithm requires the ellipsoid method for computing an m -dominating set when $k + 1 < m$. However, the ellipsoid method is not practical, so another interesting future work will be to invent a constant-approximation algorithm for the minimum weight m -dominating set problem, that does not rely on the ellipsoid method.

References

- [1] K. M. Alzoubi, P.-J. Wan, and O. Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing, MobiHoc '02*, pages 157–164, New York, NY, USA, 2002. ACM.
- [2] C. Ambühl, T. Erlebach, M. Mihalák, and M. Nunkesser. Constant-factor approximation for minimum-weight (connected) dominating sets in unit disk graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2006 and 10th International Workshop on Randomization and Computation, RANDOM 2006, Barcelona, Spain, August 28-30 2006, Proceedings*, pages 3–14, 2006.
- [3] N. Bansal and K. Pruhs. Weighted geometric set multi-cover via quasi-uniform sampling. In *Algorithms - ESA 2012 - 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings*, pages 145–156, 2012.
- [4] C. Chekuri, A. Ene, and A. Vakilian. Prize-collecting survivable network design in node-weighted graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, pages 98–109, 2012.
- [5] X. Cheng, X. Huang, D. Li, W. Wu, and D. Du. A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks. *Networks*, 42(4):202–208, 2003.
- [6] F. Cicalese, G. Cordasco, L. Gargano, M. Milanic, and U. Vaccaro. Latency-bounded target set selection in social networks. *Theoretical Computer Science*, 535:1–15, 2014.
- [7] F. Cicalese, M. Milanic, and U. Vaccaro. On the approximability and exact algorithms for vector domination and related problems in graphs. *Discrete Applied Mathematics*, 161(6):750–767, 2013.
- [8] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1-3):165–177, 1990.
- [9] F. Dai and J. Wu. On constructing k -connected k -dominating set in wireless ad hoc and sensor networks. *Journal of Parallel and Distributed Computing*, 66(7):947–958, July 2006.
- [10] T. Fukunaga. Spider covers for prize-collecting network activation problem. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 9–24, 2015.
- [11] S. Funke, A. Kesselman, U. Meyer, and M. Segal. A simple improved distributed algorithm for minimum CDS in unit disk graphs. *ACM Transactions on Sensor Networks*, 2(3):444–453, 2006.
- [12] M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. *Approximation algorithms for NP-hard problems*, pages 144–191, 1997.

- [13] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998.
- [14] D. Kim, W. Wang, X. Li, Z. Zhang, and W. Wu. A new constant factor approximation for computing 3-connected m -dominating sets in homogeneous wireless networks. In *INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA*, pages 2739–2747, 2010.
- [15] M. V. Marathe, H. Breu, H. B. H. III, S. S. Ravi, and D. J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25(2):59–68, 1995.
- [16] Z. Nutov. Approximating minimum-cost connectivity problems via uncrossable bifamilies. *ACM Transactions on Algorithms*, 9(1):1, 2012.
- [17] W. Shang, F. Yao, P. Wan, and X. Hu. On minimum m -connected k -dominating set problem in unit disc graphs. *Journal of Combinatorial Optimization*, 16(2):99–106, Dec. 2008.
- [18] M. T. Thai, N. Zhang, R. Tiwari, and X. Xu. On approximation algorithms of k -connected m -dominating sets in disk graphs. *Theoretical Computer Science*, 385(1-3):49–59, 2007.
- [19] F. Wang, M. T. Thai, and D. Du. On the construction of 2-connected virtual backbone in wireless networks. *IEEE Transactions on Wireless Communications*, 8(3):1230–1237, 2009.
- [20] W. Wang, D. Kim, M. K. An, W. Gao, X. Li, Z. Zhang, and W. Wu. On construction of quality fault-tolerant virtual backbone in wireless networks. *IEEE/ACM Transactions on Networking*, 21(5):1499–1510, 2013.
- [21] W. Wang, B. Liu, D. Kim, D. Li, J. Wang, and Y. Jiang. A better constant approximation for minimum 3-connected m -dominating set problem in unit disk graph using Tutte decomposition. In *2015 IEEE Conference on Computer Communications, INFOCOM 2015, Kowloon, Hong Kong, April 26 - May 1, 2015*, pages 1796–1804, 2015.
- [22] J. Willson, Z. Zhang, W. Wu, and D. Du. Fault-tolerant coverage with maximum lifetime in wireless sensor networks. In *2015 IEEE Conference on Computer Communications, INFOCOM 2015, Kowloon, Hong Kong, April 26 - May 1, 2015*, pages 1364–1372, 2015.
- [23] J. Yu, L. Jia, D. Yu, G. Li, and X. Cheng. Minimum connected dominating set construction in wireless networks under the beeping model. In *2015 IEEE Conference on Computer Communications, INFOCOM 2015, Kowloon, Hong Kong, April 26 - May 1, 2015*, pages 972–980, 2015.
- [24] F. Zou, X. Li, S. Gao, and W. Wu. Node-weighted Steiner tree approximation in unit disk graphs. *Journal of Combinatorial Optimization*, 18(4):342–349, 2009.
- [25] F. Zou, Y. Wang, X. Xu, X. Li, H. Du, P. Wan, and W. Wu. New approximations for minimum-weighted dominating sets and minimum-weighted connected dominating sets on unit disk graphs. *Theoretical Computer Science*, 412(3):198–208, 2011.

A Rectification of Shang et al. for the unweighted $(2, m)$ -CDS

Shang et al. [17] gave an approximation algorithm for the unweighted $(2, m)$ -CDS problem. They claimed that their algorithm achieves an approximation factor of $5 + 25/m$ for $2 \leq m \leq 5$, and 11 for $m \geq 6$. We agree with them that their algorithm achieves a constant approximation factor, but their claim contains an error. In this section, we explain which part of their claim is wrong, and we provide a correct analysis on their algorithm.

Let us begin with illustrating their algorithm. Let OPT denote the minimum size of $(2, m)$ -CDSs. Let I_i be a maximal independent set of $G[V \setminus \bigcup_{i'=0}^{i-1} I_{i'}]$ for each $i = 1, \dots, m$, where $I_0 = \emptyset$. The algorithm first computes I_1, \dots, I_m , and $C \subseteq V \setminus I_1$ such that $|C| \leq |I_1|$ and $G[C \cup I_1]$ is connected. The following properties are proven.

- (i) $|I_i| \leq \max\{5/m, 1\}\text{OPT}$ for each $i = 1, \dots, m$.
- (ii) $\bigcup_{i'=1}^i I_{i'}$ is an i -dominating set for each $i = 1, \dots, m$, and hence $T := \bigcup_{i=1}^m I_i \cup C$ is a $(1, m)$ -CDS.
- (iii) Each cut-node of $G[T]$ is included in I_1 or in C .
- (iv) $|T| \leq (5 + 5/k)\text{OPT}$ for $k \leq 5$, and $|T| \leq (7 - 5/k)\text{OPT}$ for $k \geq 6$ (this is slightly better than the conclusion in [17], but they proved this).

After this step, the algorithm computes a node set S such that $T \cup S$ is 2-connected as follows. First, S is initialized to an empty set. Then, the algorithm computes a T -cut X such that $|\Gamma_T(X)| = 1$ and no T -path in $G[T \cup S]$ covers X . For this T -cut X , the algorithm finds a T -path that covers X with at most two inner nodes, and it adds these inner nodes to the solution S . This procedure is repeated until $T \cup S$ becomes 2-connected, and the algorithm outputs $T \cup S$ as a $(2, m)$ -CDS.

Shang et al. [17] claimed that the number of iterations is at most the number of cut-nodes in $G[T]$, which is at most $|I_1| + |C| \leq 2|I_1| \leq 2\max\{5/m, 1\}\text{OPT}$, due to properties (i) and (iii). Since each iteration adds at most two nodes, when the algorithm terminates, $|S| \leq 4\max\{5/m, 1\}\text{OPT}$. This is their analysis of the algorithm.

However, the number of iterations is not bounded by the number of cut-nodes in $G[T]$. This can be observed by considering a star with n leaves. The star has only one cut-node. To make it 2-connected, we need to add $n - 1$ paths to connect the leaves.

We claim that a correct upper-bound on the number of iterations is the number of 2-connected components of $G[T]$, and this number is bounded by $|I_1| + |C| + |I_2| - 1 \leq 3\max\{5/m, 1\}\text{OPT} - 1$.

For proving our claim, let us consider the tree F that represents the 2-connected component decomposition of $G[T]$. Namely, the node set of F is the disjoint union of two node sets B and W . Each node in B corresponds to a 2-connected component of $G[T]$, and each node in W corresponds to a cut-node of $G[T]$. In what follows, we identify a node in B with the corresponding 2-connected component of $G[T]$, and we identify a node in W with the corresponding cut-node of $G[T]$. A node $b \in B$ and a node $w \in W$ are joined by an edge in F when the component b includes the cut-node w .

In each iteration of the algorithm, a T -path is selected to join two different 2-connected components of $G[T]$, and the inner nodes of the path are added to the solution S . Let u and v denote the end nodes of a T -path P , and let ρ_u be a 2-connected component that includes u . If more than one component includes u , we let ρ_u denote the one nearest to the components including v on F ; ρ_v is defined in the same way. Let x be a cut-node of $G[T]$, and let b and b' be 2-connected components that include x . When the algorithm chooses a T -path P , we add a virtual edge that joins b and b' if x , b , and b' are on the path between ρ_u and ρ_v on F .

Lemma 16. *$T \cup S$ is 2-connected if virtual edges induce a connected graph on the set of neighbors of each cut-node x in F .*

Proof. Let b_1, b_2, \dots, b_i be 2-connected components of $G[T]$ that include x , and assume that b_j and b_{j+1} are joined by a virtual edge for each $j = 1, \dots, i - 1$. Let y and y' be a node in b_1 and a node in b_i , respectively, such that $y \neq x \neq y'$. There exists a path on $G[T \cup S]$ that connects y and y' , and that does not pass through x . Hence, x is not a cut-node in $G[T \cup S]$. \square

The following lemma presents a bound on the number of iterations in this algorithm.

Lemma 17. *The number of iterations is at most $3\max\{5/m, 1\}\text{OPT} - 1$.*

Proof. Let x be a cut-node on F , and let ψ_x denote the number of connected components induced by the virtual edges on the neighbor set of x . In each iteration of the algorithm, ψ_x is decreased by at least one for some cut-node x . When the first iteration begins, ψ_x is equal to the degree of x in F . Since all leaves in F are included in B , $\sum_{x \in W} \psi_x = |B| + |W|$ holds at the beginning of the first iteration. The iterations terminate when $\sum_{x \in W} \psi_x = |W|$. Hence, the number of iterations is at most $|B|$. We will determine $|B|$ below.

Let H be a spanning tree on $G[I_1 \cup C]$. We show that each 2-connected component contains a node in $I_2 \setminus C$ or an edge in H . To see this, suppose that a component b contains no edge in H . If b contains more than one node in $I_1 \cup C$, then it includes an edge in H . Since, by assumption, this does not happen, b includes only one node in $I_1 \cup C$. Notice that each 2-connected component contains at least two nodes, because a graph with two nodes joined by an edge is 2-connected. Hence, there exists a node $v \in \bigcup_{i=2}^m I_i \setminus C$ in b . If $v \in I_2$, we are done. Suppose that $v \notin I_2$. By property (ii), v has neighbors in I_1 and in I_2 . By property (iii), v is not a cut-node in $G[T]$, so these neighbors must be inside b . If the neighbor in I_2 is included in C , b contains two nodes in $I_1 \cup C$. Since this contradicts the assumption, b contains a node in $I_2 \setminus C$.

Nodes in $\bigcup_{i=2}^m I_i \setminus C$ and edges in $G[T]$ are not contained in more than one 2-connected component of $G[T]$. The number of edges in H is $|I_1| + |C| - 1 \leq 2 \max\{5/m, 1\} \text{OPT} - 1$, and $|I_2| \leq \max\{5/m, 1\} \text{OPT}$ by property (i). Hence $|B| \leq 3 \max\{5/m, 1\} \text{OPT} - 1$. \square

From Lemma 17 and property (iv), we obtain the following approximation guarantee for the algorithm.

Theorem 4. *The algorithm of Shang et al. [17] attains an approximation factor $5 + 35/k$ for $2 \leq k \leq 5$, and $13 - 5/k$ for $k \geq 6$.*